# Assignment 2 Report

CS6400 Spring ‘17 Team 73

**Modifications From Project 1** 1

**EER to Relational Mapping** 3

**SQL Create Table Statements** 3

**Tasks with SQL** 9

Login 9

View Site Data 10

Edit Site Data 11

Create Request 12

Edit Request 12

Approve Request 13

View Requests Outstanding 14

View Request Status 14

Search Clients 15

View Client Data 16

View Waitlist 16

Delete Waitlist Item 17

Move Waitlist Item 17

Search Inventory Items 18

Change Inventory Count 19

Add Inventory Item 19

View Meals 20

View Bunks/Rooms 21

Add Client 22

Check In Client to Service 23

Modify Bunk Count 24

**SQL code to create schema** 24

**Appendix 1: SQL test code** 29

# Modifications From Project 1

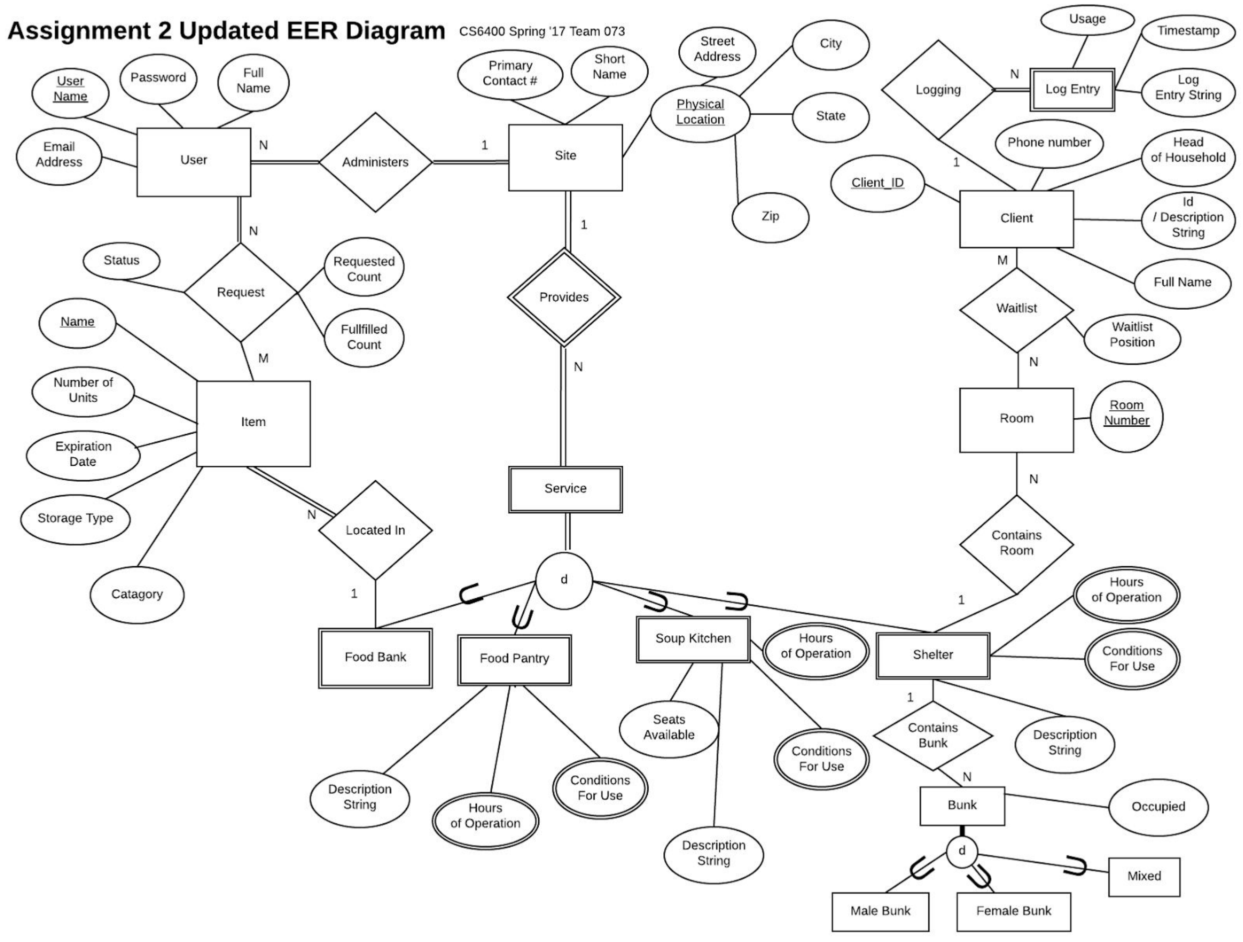
**Modifications Based on Phase 1 Feedback:**

- Made log entry a weak entity
- Made item name a key
- Added ID to client, distinct from client ID
- Corrected notation for request relationship
- Corrected cardinality between site and service
- Made relationship between user and site mandatory
- Made relationship between log and client mandatory on the log side.
- Added phone number to client entity

Additional Modifications

- Team decided to remove the unnecessary relationship 'Usage of' between Client and Services. This will be captured in the log entry as per further interpretation of the requirements document.

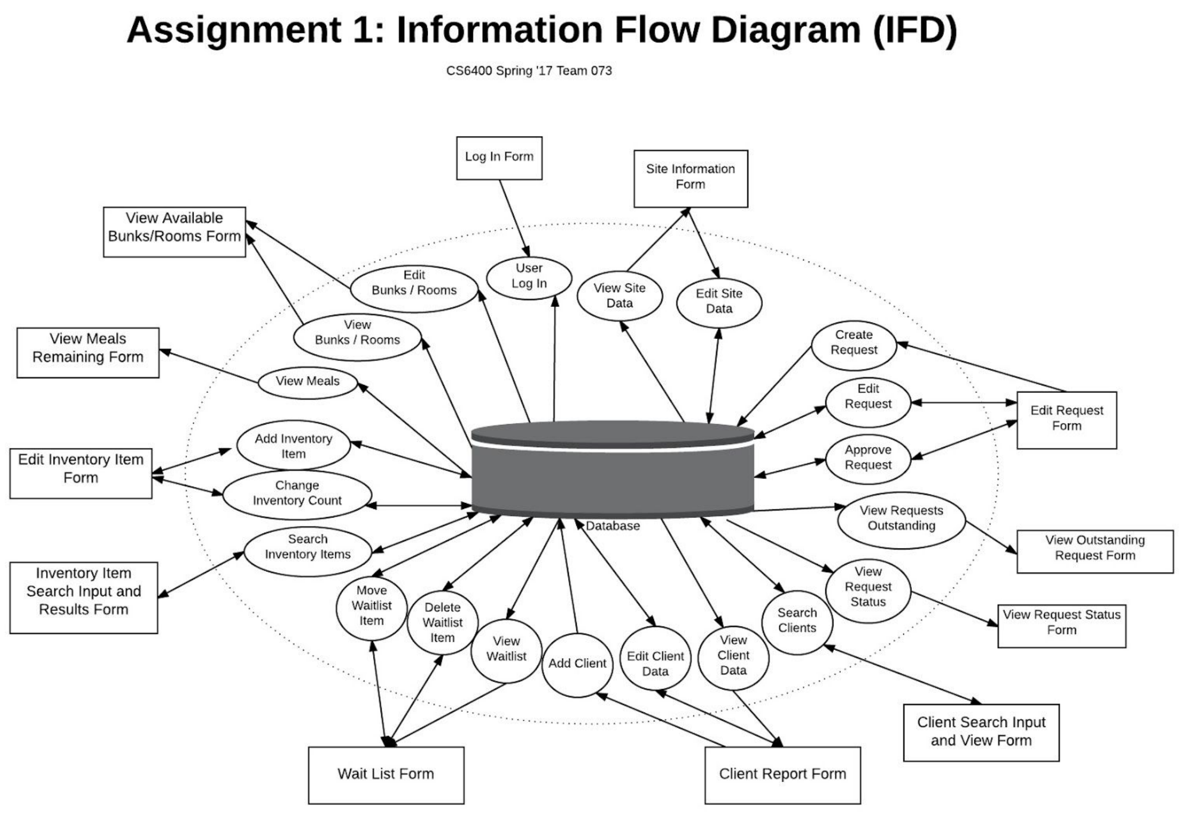
**Updated EER:**



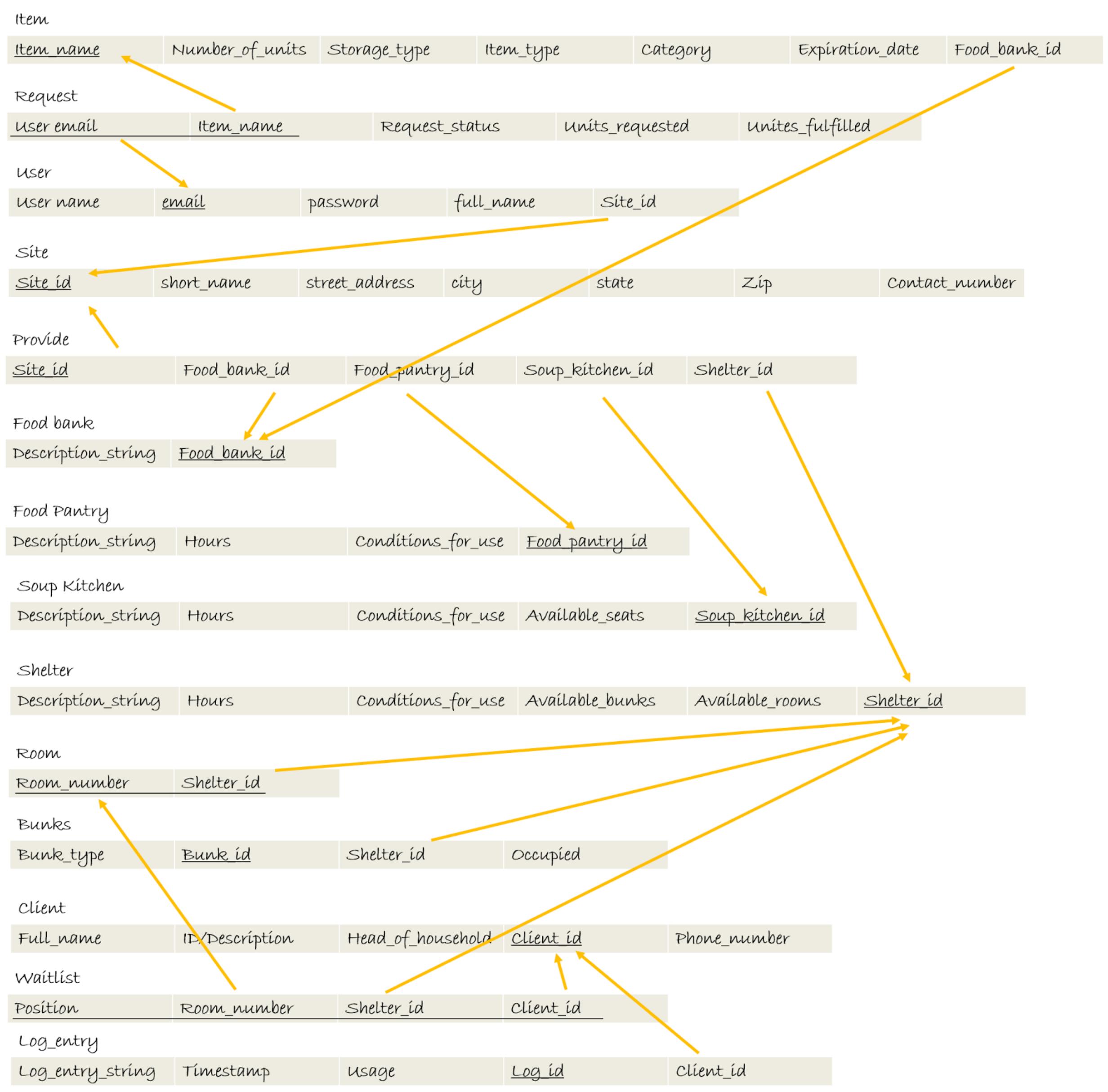
## IFD Modifications Based on Phase 1 feedback:

- Added new tasks add/enroll client, modify bunk counts
- Separated forms for outstanding request report and status. View outstanding request task should corresponds to outstanding request report, view requests status task should corresponds to request status report. Don't group into a single form.

## Updated IFD:



# EER to Relational Mapping



Item

| | | | | | | |
|---|---|---|---|---|---|---|
| Item_name | Number_of_units | Storage_type | Item_type | Category | Expiration_date | Food_bank_id |

Request

| | | | | |
|---|---|---|---|---|
| User email | Item_name | Request_status | Units_requested | Unites_fulfilled |

User

| | | | | |
|---|---|---|---|---|
| User name | email | password | full_name | Site_id |

Site

| | | | | | | |
|---|---|---|---|---|---|---|
| Site_id | short_name | street_address | city | state | Zip | Contact_number |

Provide

| | | | | |
|---|---|---|---|---|
| Site_id | Food_bank_id | Food_pantry_id | Soup_kitchen_id | Shelter_id |

Food bank

| | |
|---|---|
| Description_string | Food_bank_id |

Food Pantry

| | | | |
|---|---|---|---|
| Description_string | Hours | Conditions_for_use | Food_pantry_id |

Soup Kitchen

| | | | | |
|---|---|---|---|---|
| Description_string | Hours | Conditions_for_use | Available_seats | Soup_kitchen_id |

Shelter

| | | | | | |
|---|---|---|---|---|---|
| Description_string | Hours | Conditions_for_use | Available_bunks | Available_rooms | Shelter_id |

Room

| | |
|---|---|
| Room_number | Shelter_id |

Bunks

| | | | |
|---|---|---|---|
| Bunk_type | Bunk_id | Shelter_id | Occupied |

Client

| | | | | |
|---|---|---|---|---|
| Full_name | ID/Description | Head_of_household | Client_id | Phone_number |

Waitlist

| | | | |
|---|---|---|---|
| Position | Room_number | Shelter_id | Client_id |

Log_entry

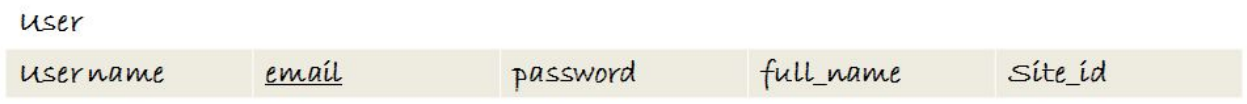
| | | | | |
|---|---|---|---|---|
| Log_entry_string | Timestamp | Usage | Log_id | Client_id |

## SQL Create Table Statements

(In format of lecture notes. For .sql file used to create mySQL database, see final section)

### USER

```
CREATE TABLE ‘User’(
        username      varchar(250)   NOT NULL,
        email         varchar(250)   NOT NULL,
        password      varchar(50)    NOT NULL,
        full_name     varchar(250)   NOT NULL,
        site_id       integer        NULL,
        PRIMARY_KEY(email),
        FOREIGN_KEY(site_id))
                REFERENCES Site (site_id) );
```

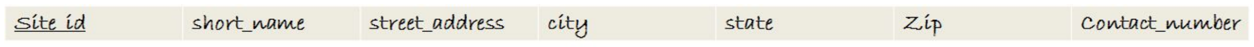


User

| username | <u>email</u> | password | full_name | Site_id |
|---|---|---|---|---|

### SITE

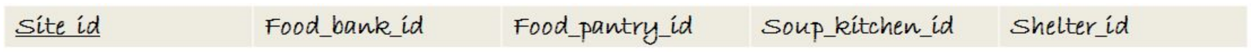
```
CREATE TABLE ‘Site’(
        Site_id                 integer        NOT NULL,
        short_name              varchar(250)   NOT NULL,
        street_address          varchar(250)   NOT NULL,
        city                    varchar(250)   NOT NULL,
        state                   varchar(50)    NOT NULL,
        zip                     integer        NOT NULL,
        contact_number          varchar(50)    NULL,
        PRIMARY_KEY(Site_id));
```



| <u>Site id</u> | short_name | street_address | city | state | Zip | Contact_number |
|---|---|---|---|---|---|---|

**PROVIDE**

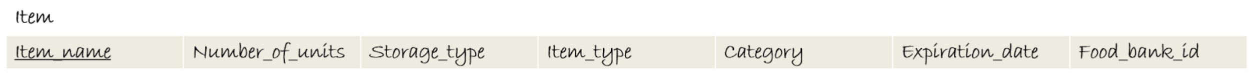
```
CREATE TABLE 'Provide'(
        site_id                 integer     NOT NULL,
        food_bank_id            integer     NULL,
        food_pantry_id          integer     NULL,
        soup_kitchen_id         integer     NULL,
        shelter_id              integer     NULL,
        PRIMARY_KEY(site_id)),
        FOREIGN_KEY(site_id))
                REFERENCES Site (site_id),
        FOREIGN_KEY(food_bank_id))
                REFERENCES Food_Bank (food_bank_id),
        FOREIGN_KEY(food_pantry_id))
                REFERENCES Food_Pantry (food_pantry_id),
        FOREIGN_KEY(soup_kitchen_id))
                REFERENCES Soup_Kitchen (soup_kitchen_id),
        FOREIGN_KEY(shelter_id))
                REFERENCES Shelter (shelter_id));
```



| Site_id | Food_bank_id | Food_pantry_id | Soup_kitchen_id | Shelter_id |
|---|---|---|---|---|

**ITEM**

```
CREATE TABLE 'Item'(
        item_name                       varchar(250)  NOT NULL,
        number_of_units                 integer       NULL,
        storage_type                    enum          NULL,
        item_type                       enum          NULL,
        food_category                   enum          NULL,
        supply_catagory                 enum          NULL,
        expiration_date                 DATETIME      NULL,
        food_bank_id                    integer       NULL,
        PRIMARY_KEY(item_name)),
        FOREIGN_KEY(food_bank_id))
                REFERENCES Food_Bank (food_bank_id));
```

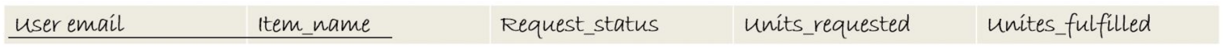


Item

| Item_name | Number_of_units | Storage_type | Item_type | Category | Expiration_date | Food_bank_id |
|---|---|---|---|---|---|---|

**REQUEST**

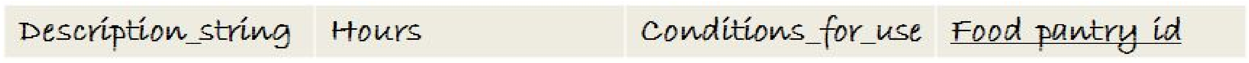
```
CREATE TABLE ‘Request’(
        email                                       varchar(250)   NOT NULL,
        item_name                                   varchar(250)   NOT NULL,
        request_status                              integer        NULL,
        units_requested                             integer        NULL,
        units_fulfilled                             integer        NULL,
        PRIMARY_KEY(email, item_name)),
        FOREIGN_KEY(email))
                REFERENCES User (email)),
        FOREIGN_KEY(item_name))
                REFERENCES Item (item_name));
```



| | | | | |
|---|---|---|---|---|
| User email | Item_name | Request_status | Units_requested | Unites_fulfilled |

**FOOD PANTRY**

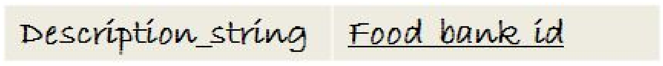
```
CREATE TABLE ‘Food Pantry’(
        food_pantry_id              integer        NOT NULL,
        Description_string          varchar(250)   NOT NULL,
        Hours                       varchar(50)    NOT NULL,
        Conditions_for_use          varchar(250)   NOT NULL,
        PRIMARY_KEY(food_pantry_id)) );
```



| | | | |
|---|---|---|---|
| Description_string | Hours | Conditions_for_use | Food pantry id |

**FOOD BANK**

```
CREATE TABLE ‘Food Bank’(
        Food_bank_id                integer        NOT NULL,
        Description_string          varchar(50)    NOT NULL,
        PRIMARY_KEY(Food_bank_id)) );
```



| | |
|---|---|
| Description_string | Food bank id |

**SOUP KITCHEN**

```
CREATE TABLE ‘Soup Kitchen’(
        soup_kitchen_id             integer        NOT NULL,
        Description_string          varchar(250)   NOT NULL,
        Hours                       varchar(50)    NOT NULL,
        Conditions_for_use          varchar(50)    NOT NULL,
        available_seats             integer        NOT NULL,
        PRIMARY_KEY(soup_kitchen_id)) );
```

| | | | | |
|---|---|---|---|---|
| Description_string | Hours | Conditions_for_use | Available_seats | Soup kitchen id |

**SHELTER**

```
CREATE TABLE 'Shelter'(
        Shelter_id                  integer        NOT NULL,
        Description_string          varchar(250)   NOT NULL,
        Hours                       varchar(250)   NOT NULL,
        Conditions_for_use          varchar(250)   NOT NULL,
        available_bunks             integer        NOT NULL,
        available_rooms             integer        NOT NULL,
        PRIMARY_KEY(Shelter_id)) );
```

| | | | | | |
|---|---|---|---|---|---|
| Description_string | Hours | Conditions_for_use | Available_bunks | Available_rooms | Shelter id |

**ROOM**

```
CREATE TABLE 'Room'(
        room_number                 integer        NOT NULL,
        Shelter_id                  integer        NOT NULL,
        PRIMARY_KEY(room_number,Shelter_id)),
        FOREIGN_KEY(Shelter_id))
                REFERENCES Shelter (Shelter_id));
```

| | |
|---|---|
| Room_number | Shelter_id |

**BUNK**

```
CREATE TABLE 'Bunk'(
        bunk_type                   enum           NOT NULL,
        bunk_id                     integer        NOT NULL,
        Shelter_id                  integer        NOT NULL,
        occupied                    boolean        NOT NULL,
        PRIMARY_KEY(bunk_id)),
        FOREIGN_KEY(Shelter_id))
                REFERENCES Shelter (Shelter_id));
```

| | | | |
|---|---|---|---|
| Bunk_type | Bunk_id | Shelter_id | Occupied |

**CLIENT**

```
CREATE TABLE ‘Client’(
        client_id                               integer         NOT NULL,
        full_name                               varchar(250)    NOT NULL,
        description_string                      varchar(250)    NOT NULL,
        head_of_household                       boolean         NOT NULL,
        phone_number                            varchar(50),
        PRIMARY_KEY(client_id)));
```

Client

| Full_name | ID/Description | Head_of_household | Client_id | Phone_number |
|---|---|---|---|---|

**WAITLIST**

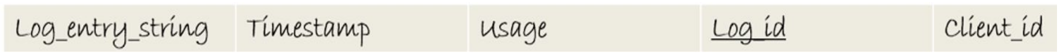
```
CREATE TABLE ‘Waitlist’(
        position                                integer         NOT NULL,
        room_number                             integer         NOT NULL,
        shelter_id                              integer         NOT NULL,
        client_id                               integer         NOT NULL,
        PRIMARY_KEY(position, room_number, shelter_id, client_id))
        FOREIGN_KEY(room_number))
                REFERENCES Room (room_number));
        FOREIGN_KEY(shelter_id))
                REFERENCES Client (shelter_id));
        FOREIGN_KEY(client_id))
                REFERENCES Client (client_id));
```

Waitlist

| Position | Room_number | Shelter_id | Client_id |
|---|---|---|---|

**LOG ENTRY**

```
CREATE TABLE ‘Log_entry’(
        log_id                                  integer         NOT NULL,
        log_entry_string                        varchar(250)    NOT NULL,
        timestamp                               DATETIME        NOT NULL,
        usage                                   integer         NOT NULL,
        client_id                               integer         NOT NULL,
        PRIMARY_KEY(log_id)),
        FOREIGN_KEY(client_id))
                REFERENCES Client (client_id));
```



| Log_entry_string | Timestamp | Usage | Log_id | Client_id |
|---|---|---|---|---|

# Tasks with SQL

## Login

**Task Decomposition:**

**Lock Types**: Read-only on User table
**Enabling Conditions:** None

**Frequency**: Frequent
**Schemas**: Single
**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
SELECT count(username) FROM User WHERE username=$username and password=$password;

- User enters *username* ($Username), *password* ($Password) input fields.
- If data validation is successful for both *username* and *password* input fields, then when **Enter** button is clicked:
- If User record is found but User.password != '$Password or User record is not found':
    - Go back to ***Login*** form, with error message.

## View Site Data

**Task Decomposition:**

**Lock Types**: Read only on site table, services tables
**Enabling Conditions:** None, publicly available
**Frequency**: Frequent
**Schemas**: Site table, services table
**Consistency:** Not important

**Subtasks:** Requires mother task, but all tasks can be done in parallel. Display site in top frame, display each service in separate frame

**Abstract Code:**
SELECT Site.short_name, Site.street_address, Site.city, Site.state, Site.full_name, Site.zip, Site.contact_number,Food_Pantry.description_string, Food_Pantry.hours, Food_Pantry.conditions_for_use, Food_Bank.description_string, Soup_Kitchen.description_string, Soup_Kitchen.hours, Soup_Kitchen.conditions_for_use, Soup_Kitchen.available_seats, Shelter.description_string, Shelter.hours, Shelter.conditions_for_use, Shelter.available_bunks, Shelter.available_rooms FROM Site LEFT JOIN Provide on Provide.site_id=Site.site_id LEFT JOIN Food_Pantry on Food_Pantry.food_pantry_id=Provide.food_pantry_id LEFT JOIN Food_Bank on Food_Bank.food_bank_id=Provide.food_bank_id LEFT JOIN Soup_Kitchen on Soup_Kitchen.soup_kitchen_id=Provide.soup_kitchen_id LEFT JOIN Shelter on Shelter.shelter_id=Provide.shelter_id WHERE Site.site_id=$site_id;

- User enters site_id
- Display all attributes of the site, or any service the site proves

## Edit Site Data

**Task Decomposition:**
**Lock Types**: Read/write on site/service table
**Enabling Conditions:** Logged in as user who administers site
**Frequency**: Infrequent, but more common to modify service than modify site
**Schemas**: Site table, services tables, attributes of any sub-class
**Consistency:** Important

**Subtasks:**Mother Task is not needed. No decomposition needed.

**Abstract Code:**
UPDATE Site set short_name=$short_name,street_address=$street_addr,city=$city,state=$state,full_name=$name,zip=$zip,contact_number=$contact WHERE site_id=$site_id;

- Display ***View Site Data***
- Display dropdown for service type
- Display text field for all *attributes* of any service
- Display ***X*** button next to each service
- When ***Add Service*** Button is pressed:
    - Validate input
    - Create new service with displayed attributes
- When ***Submit*** Button is pressed:
    - Write attributes of site

## Create Request

**Task Decomposition:**
**Lock Types** Write lock on request table, read on request_items
**Enabling Conditions:** Logged in as user

**Frequency**: Frequent
**Schemas**: Request table, keys from user and item tables
**Consistency:** Important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
INSERT INTO Request(username,item_name,
request_status,units_requested,units_fulfilled) VALUES
($username,$requested_item,"pending",$requested_count,0);

- Display ***View Items*** form
- Display the $*item count* text field next to each item
- When ***Submit*** button is pressed:
    - Validate input (e.g. If $request is for food bank at $users site)
    - Create request of selected item, from $user, with given count

## Edit Request

**Task Decomposition:**

**Lock Types**: Read/Write lock on request table
**Enabling Conditions:** Logged in as user

**Frequency**: Frequent
**Schemas**: Request table
**Consistency:** Important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
UPDATE Request SET Request.units_requested=$units_requested WHERE Request.username=$requestingUser AND Request.item_name=$item_name;

- Display ***View Requests*** form
- When the ***Submit*** button is pressed:
    - Validate input
    - If $user is not original requester
        - Error
    - Else:
        - Set count of displayed request to updated value.

## Approve Request

**Task Decomposition:**
**Lock Types**: Read/write lock on request table, read/write lock on inventory table
**Enabling Conditions:** Logged in as user for food bank
**Frequency**: Frequent
**Schemas**: Requests, items
**Consistency:** Important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**

UPDATE Request INNER JOIN Item on Request.item_name=Item.item_name INNER JOIN User on Item.food_bank_id=User.site_id SET Request.units_fulfilled=$fulfilled_count, Request.request_status="approved", Item.number_of_units=Item.number_of_units-$fulfilled_count WHERE User.username=$username AND Request.username=$requestingUser AND Request.item_name=$itemName AND Item.number_of_units>$fulfilled_count;

- Display ***View Requests*** outstanding Form
- Display *approved amount* attribute next to each request associated with $user's food bank (if any)
- When ***Approve*** button is pressed:
    - If item count at food bank is less than or equal to approved amount
    - Reduce item count at food bank by approved amount
    - Set request status to resolved
    - Else:
        - Error

## View Requests Outstanding

**Task Decomposition:**

**Lock Types**: Read on user table, read on sites table, read on requests table, read on inventory table

**Enabling Conditions:** Logged in as user for food bank
**Frequency**: Frequent
**Schemas**: Requests, sites, users, inventory
**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
SELECT Request.username, Request.item_name, Request.request_status, Request.units_requested, Request.units_fulfilled FROM Request INNER JOIN Item on Request.item_name=Item.item_name INNER JOIN User on Item.food_bank_id=User.site_id WHERE User.username=$username;

- Find $site administered by $user
- For each $request associated with $site
    - Display $request.user, $request.requested count

  - Display $count of corresponding $item at food bank (0 if there is no matching item)
  - Calculate the total number of that item that have been requested
  - If the total requests for that item exceed the supply, highlight the line in red

Note: Site must have a food bank according to enabling conditions to display requests associated with a given food bank.

## View Request Status

**Task Decomposition:**
**Lock Types**: Read on user table, read on requests table
**Enabling Conditions:** Logged in as user

**Frequency**: Frequent
**Schemas**: Requests, users
**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
SELECT username, item_name, request_status,units_requested,units_fulfilled FROM Request WHERE username= $username ORDER BY request_status;

- Open all requests associated with $user
- For each $request ordered by status
  - Display $request.date, $request.site, $request.requested_count, $request.approved count, $request.status

## Search Clients

**Task Decomposition:**
**Lock Types**: Read on clients
**Enabling Conditions:** Logged in as user

**Frequency**: Frequent
**Schemas**: Single
**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
SELECT full_name, client_id, description_string, head_of_household FROM Client WHERE full_name like "%$search_field%";

- Display text box $*input*
- On ***Search*** button press:
    - Validate the search text box
    - $client_list= Find all clients with name like %$input%
    - If count($client_list)>5, error("Please enter more unique search criteria")
    - Else
        - For each $client
            - Display $client.id, $client.name, $client.head of household status

## View Client Data

**Task Decomposition:**
**Lock Types**: Read on clients
**Enabling Conditions:** Logged in as user

**Frequency**: Frequent
**Schemas**: Single
**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
SELECT full_name, description_string, head_of_household FROM Client WHERE client_id=$client_id;

- Display text box $*input*
- On ***View Client*** button press:
    - Validate input
        - Find client with identifier $input
        - Display $client.name, $client.id, and $client.head_of_household

## View Waitlist

**Task Decomposition:**
**Lock Types**: Read on waitlist, read on user, read on site, read on client

**Enabling Conditions:** Logged in as user with shelter
**Frequency**: Frequent
**Schemas**: Single
**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**

SELECT client_id, position from Waitlist WHERE shelter_id=$shelter_number AND room_number=$room_number ORDER BY position ASC;

- Determine shelter associated with site associated with $user
- For each $client on waitlist, sorted by waitlist position
  - Display $client

## Delete Waitlist Item

**Task Decomposition:**
**Lock Types**: Read/write on waitlist
**Enabling Conditions:** Logged in as user associated with shelter
**Frequency**: Infrequent
**Schemas**: Multiple
**Consistency:** Important

**Subtasks:** Mother Task is not needed. No decomposition needed.
DELETE FROM Waitlist WHERE position=$position and shelter_id=$shelter_id AND room_number=$room_number;
UPDATE Waitlist SET position=position-1 WHERE position>$position AND shelter_id=$shelter_id AND room_number=$room_number;

**Abstract Code:**

- Display ***Waitlist*** form
- Display ***X*** button next to each waitlist item
- On '***X***' button press:
  - Set $previous_position to $selected_item.position
  - Remove $selected_item
  - For each $waitlist_item with position above $selected_item.position, decrement $selected_item.position

## Move Waitlist Item

**Task Decomposition:**
**Lock Types**: Read/write on waitlist
**Enabling Conditions:** Logged in as user associated with shelter
**Frequency**: Infrequent
**Schemas**: Multiple
**Consistency:** Important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
UPDATE Waitlist SET position=position-1 WHERE position>$old_position AND shelter_id=$shelter_id AND room_number=$room_number ;
UPDATE Waitlist SET position=position+1 WHERE position>$new_position AND shelter_id=$shelter_id AND room_number=$room_number;

- Display ***Waitlist*** Form
- Display text box1 ($*selected_item*)
- Display text box1 ($*New_position*)
- On ***'(Up Arrow)'*** or '(***Down Arrow*** )' button press (up arrow and down arrow are represented as arrow icons):
    - Validate input (e.g. new position >0 and <len(waitlist))
    - Set $previous_position to $selected_item.position
    - Set $selected_item.position to $new_position
    - For each $waitlist_item with position above $previous_position
        - Decrement $selected_item.position
    - For each $waitlist_item with position below $selected_item.position:
        - Increment $selected_item.position

## Search Inventory Items

**Task Decomposition:**
**Lock Types**: Read on inventory, read on site, read on user
**Enabling Conditions:** Logged in as user

**Frequency**: Frequent
**Schemas**: Multiple
**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
SELECT item_name, number_of_units, storage_type, item_type, food_category, supply_category,expiration_date, food_bank_id FROM Item where ($expiration_Date="*" or expiration_date=$expiration_date) AND ($stoarge_type="*" OR storage_type=$storage_type) AND ($food_type="*" OR item_type=$food_type) AND ($food_category="*" OR food_category=$food_category") AND($supply_category="*" OR supply_category=$supply_category) AND item_name LIKE "%$item_name%";

- Display a text box for $*expiration date* (default ‘’)
- Display a dropdown for $*storage_type* (containing all known storage types)
- Display a dropdown for $*type* (containing ‘food’ and ‘supply’)
- Display a dropdown for $*category* (containing initially all categories of food and supply)
- Display a text box for $*keyword* (default ‘’)
- On dropdown select:
    - Select all inventory matching new restrictions.

## Change Inventory Count

**Task Decomposition:**
**Lock Types**: Read/write on inventory, read on users, read on site
**Enabling Conditions:** Logged in as user associated with food bank
**Frequency**: Moderate
**Schemas**: Multiple
**Consistency:** Important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**

UPDATE Item SET number_of_units=$NUM_UNITS WHERE item_name=$ITEM_NAME;
UPDATE Request  INNER JOIN Item on Item.item_name=Request.item_name SET request_status=”closed”, units_fulfilled=0 WHERE item_name=$ITEM_NAME AND Item.item_name=$ITEM_NAME and Item.number_of_units=0

- Display ***Search Inventory*** item form.
- Display text field $*count* next to each found item
- On ***Search*** button press:
    - For $item with non-empty $field value

        - If $item not in $users food bank
            - Display error, continue
        - Else:
            - $item.count=$count

## Add Inventory Item

**Task Decomposition:**
**Lock Types**: Write on inventory, read on user, read on site
**Enabling Conditions:** Logged in as user associated with food bank
**Frequency**: Moderate
**Schemas**: Multiple
**Consistency:** Important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
SELECT distinct(item_type) FROM Item;
SELECT distinct(food_category) FROM Item;
INSERT INTO Item(item_name, number_of_units, storage_type, item_type, food_category,supply_category, expiration_date, food_bank_id) VALUES ($ITEM_NAME, $NUM_UNITS, $STORAGE_TYPE, $ITEM_TYPE, $FOOD_CATEGORY, $SUPPLY_CATEGORY, $EXPIRATION_DATE, $FOOD_BANK_ID);

- Display dropdown for $*item_type* (Containing all available item types)
- Display dropdown for $*item_category* .
- Display text box for $*expiration_date* (default 01/01/9999)
- Display text box for $*description*.
- On button press:
    - Validate input
    - Create new item with the site associated with the user, attributes according to input fields
    - Reset fields

## View Meals

**Task Decomposition:**

**Lock Types**: Read on items
**Enabling Conditions:** None

**Frequency**: Moderate
**Schemas**: Single
**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
SELECT min(counts.count) as low, max(counts.count) AS total_meals FROM (
SELECT 'vegetable' AS type, count(item_name) AS count FROM Item WHERE
food_category = 'vegetables'
UNION
SELECT 'mineral' AS type, count(item_name) AS count FROM Item WHERE
food_category = 'beans' OR food_category = 'nuts' OR food_category = 'grains'
UNION
SELECT 'animal' AS type, count(item_name) AS count FROM Item WHERE
food_category = 'meat' OR food_category = 'seafood' OR food_category = 'dairy') AS
counts;
SELECT counts.type as send_more FROM (
SELECT 'vegetable' AS type, count(item_name) AS count FROM Item WHERE
food_category = 'vegetables'
UNION
SELECT 'mineral' AS type, count(item_name) AS count FROM Item WHERE
food_category = 'beans' OR food_category = 'nuts' OR food_category = 'grains'
UNION
SELECT 'animal' AS type, count(item_name) AS count FROM Item WHERE
food_category = 'meat' OR food_category = 'seafood' OR food_category = 'dairy') AS
counts WHERE counts.count=$low;

- $v= sum(count) of $items where $item.category= vegetables
- $n= sum(count) of $items where $item.category= nuts or $item.category = grains or $item.category = beans
- $p= sum(count) of $items where $item.category= meat or $item.category = seafood or $item.category = dairy or $item.category = eggs
- Display “Total meals =”+min($v,$n,$p)
- Display “Need more “+argmin($v, $n, $p)

## View Bunks/Rooms

**Task Decomposition:**
**Lock Types**: Read on site, rooms
**Enabling Conditions:** None

**Frequency**: Frequent
**Schemas**: Mixed

**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**

SELECT count(Bunk.bunk_number), bunk_type, Shelter.description_string, Shelter.hours, Shelter.conditions_for_use FROM Bunk INNER JOIN Shelter on Shelter.shelter_id=Bunk.shelter_id GROUP BY Bunk.bunk_type, Shelter.shelter_id

```
//If empty, "Sorry, all shelters are currently at maximum capacity"
    ● $found = false
    ● For each $shelter:
        ○ $male=$shelter.male_bunks
        ○ $female=$shelter.female_bunks
        ○ $mixed=$shelter.mixed_bunks
        ○ If $mixed = 0 and $male = 0 and $female = 0:
            ■ Continue
        ○ $found= true
        ○ Display $shelter.name
        ○ Display $shelter.location
        ○ Display $shelter.phone_number
        ○ Display $shelter.hour_of_operations
        ○ Display $shelter.conditions
        ○ Display $bunks
●   If not $found:
        ○ Display "Sorry, all shelters are currently at maximum capacity"
```

## Add Client

**Task Decomposition:**
**Lock Types**: Write on client
**Enabling Conditions:** None

**Frequency**: Moderate
**Schemas**: Single

**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
Insert into Client(full_name, description_string, head_of_household)
VALUES($FullName,$Description,$HeadOfHousehold);

- Display fields for $FullName, $ID/Description, $HeadOfHouseHold, $PhoneNumber
- On Button Press:
    - Insert into clients client with $Unique ID,$FullName, $ID/Description, $HeadOfHouseHold, $PhoneNumber

## Check In Client to Service

**Task Decomposition:**
**Lock Types**: Read/Write on Shelter, Read on User, Write on log
**Enabling Conditions:** Logged in as user for a service

**Frequency**: Moderate
**Schemas**: Single

**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
SELECT Shelter.shelter_id, Bunk.bunk_number from Shelter INNER JOIN User on User.site_id=Shelter.shelter_id INNER JOIN Bunk on Bunk.shelter_id=Shelter.shelter_id WHERE Shelter.available_bunks>0 AND User.username = $username AND Bunk.bunk_type=$gender OR Bunk.bunk_type="mixed" ORDER BY Bunk.bunk_type LIMIT 1;

UPDATE Shelter INNER JOIN Bunk on Bunk.shelter_id=Shelter.shelter_id SET Shelter.available_rooms=Shelter.available_bunks-1,Bunk.occupied=True WHERE Shelter.shelter_id=$shelter_id AND Bunk.bunk_number=bunk_number AND $service_type="shelter";

INSERT INTO Log_Entry(log_entry_string, timestamp, log_usage, client_id) values($log_entry,now(), $usage, $client_id);

- Request $gender
- Identify service associated with $user
- If service has available bunks:
    - Find bunk associated with service which is not occupied and with gender $gender
    - If none, find bunk associated with service which is not occupied and with gender $mixed
    - If not none:
        - Set bunk to occupied
        - Decrease bunk_count by 1

## Modify Bunk Count

**Task Decomposition:**
**Lock Types**: Read/Write on Shelter, Read on User
**Enabling Conditions:** Logged in as user for a service

**Frequency**: Infrequent
**Schemas**: Single

**Consistency:** Not important

**Subtasks:** Mother Task is not needed. No decomposition needed.

**Abstract Code:**
Update Shelter INNER JOIN User ON User.site_id=Shelter.shelter_id SET Shelter.available_bunks=$available_bunks WHERE User.username=$username

- Request $updated_bunk_count
- Identify shelter associated with $user

- If $updated_bunk_count>0:
    - Update shelter.bunk_count to $updated_bunk_count

## SQL code to create schema

```
DROP DATABASE IF EXISTS `cs6400_sp17_team073`;


SET default_storage_engine=InnoDB;


CREATE DATABASE IF NOT EXISTS cs6400_sp17_team073 DEFAULT CHARACTER SET
utf8 COLLATE utf8_general_ci;
USE cs6400_sp17_team073;

-- Tables

CREATE TABLE User (
  username varchar(250) NOT NULL,
  user_email varchar(250) NOT NULL,
  password varchar(50) NOT NULL,
  full_name varchar(250) NOT NULL,
  site_id int(16)unsigned,
  PRIMARY KEY (username)
);

CREATE TABLE Site (
  site_id int(16) unsigned NOT NULL AUTO_INCREMENT,
  short_name varchar(250) NOT NULL,
  street_address varchar(250) NOT NULL,
  city varchar(250) NOT NULL,
  state varchar(50) NOT NULL,
  full_name varchar(250) NOT NULL,
  zip int(16) unsigned NOT NULL,
  contact_number varchar(50) NOT NULL,
  PRIMARY KEY (site_id)
);
```

```
CREATE TABLE Provide (
  site_id int(16) unsigned NOT NULL AUTO_INCREMENT,
  food_bank_id int(16) unsigned,
  food_pantry_id int(16) unsigned,
  soup_kitchen_id int(16) unsigned,
  shelter_id int(16) unsigned,
  PRIMARY KEY (site_id)
);

CREATE TABLE Item (
  item_name varchar(250) NOT NULL,
  number_of_units int(16) unsigned,
  storage_type int(16) unsigned NOT NULL,
  item_type int(16) unsigned NOT NULL,
  food_category int(16) unsigned NOT NULL,
  supply_category int(16) unsigned NOT NULL,
  expiration_date DATETIME NOT NULL,
  food_bank_id int(16) unsigned,
  PRIMARY KEY (item_name)
);

CREATE TABLE Request (
  username varchar(250) NOT NULL,
  item_name varchar(250) NOT NULL,
  request_status int(16) unsigned NOT NULL,
  units_requested int(16) unsigned NOT NULL,
  units_fulfilled int(16) unsigned,
  PRIMARY KEY (username,item_name)
);

CREATE TABLE Food_Pantry (
  food_pantry_id int(16) unsigned NOT NULL AUTO_INCREMENT,
  description_string varchar(250) NOT NULL,
  hours varchar(50) NOT NULL,
  conditions_for_use varchar(250) NOT NULL,
  PRIMARY KEY (food_pantry_id)
);


CREATE TABLE Food_Bank (
  food_bank_id int(16) unsigned NOT NULL AUTO_INCREMENT,
  description_string varchar(250) NOT NULL,
  PRIMARY KEY (food_bank_id)
);

CREATE TABLE Soup_Kitchen (
  soup_kitchen_id int(16) unsigned NOT NULL AUTO_INCREMENT,
  description_string varchar(250) NOT NULL,
  hours varchar(50) NOT NULL,
  conditions_for_use varchar(250) NOT NULL,
  available_seats int(16),
  PRIMARY KEY (soup_kitchen_id)
);

CREATE TABLE Shelter (
  shelter_id int(16) unsigned NOT NULL AUTO_INCREMENT,
  description_string varchar(250) NOT NULL,
  hours varchar(50) NOT NULL,
  conditions_for_use varchar(250) NOT NULL,
  available_bunks int(16),
  available_rooms int(16),
  PRIMARY KEY (shelter_id)
);

CREATE TABLE Room (
  room_number int(16) unsigned NOT NULL AUTO_INCREMENT,
  shelter_id int(16) unsigned NOT NULL,
  PRIMARY KEY (room_number,shelter_id)
);

CREATE TABLE Bunk (
  bunk_number int(16) unsigned NOT NULL AUTO_INCREMENT ,
  bunk_type int(16) unsigned NOT NULL,
  shelter_id int(16) unsigned NOT NULL,
  occupied boolean,
  PRIMARY KEY (bunk_number)
);

CREATE TABLE Client (
  client_id int(16) unsigned NOT NULL AUTO_INCREMENT,
  full_name varchar(250) NOT NULL,
  description_string varchar(250) NOT NULL,
  head_of_household boolean,
  PRIMARY KEY (client_id)
);
```

```
CREATE TABLE Waitlist (
  position int(16) unsigned NOT NULL,
  room_number int(16) unsigned NOT NULL,
  shelter_id int(16) unsigned NOT NULL,
  client_id int(16) unsigned NOT NULL,
  PRIMARY KEY (position,room_number,client_id,shelter_id)
);

CREATE TABLE Log_Entry (
  log_id int(16) unsigned NOT NULL AUTO_INCREMENT,
  log_entry_string varchar(250) NOT NULL,
  timestamp datetime NOT NULL,
  log_usage int(16) unsigned NOT NULL,
  client_id int(16) unsigned NOT NULL,
  PRIMARY KEY (log_id)
);


--  Table Constraints

ALTER TABLE `User`
  ADD CONSTRAINT User_ibfk_1 FOREIGN KEY (site_id) REFERENCES `Site` (site_id);

ALTER TABLE `Provide`
  ADD CONSTRAINT Provide_ibfk_1 FOREIGN KEY (site_id) REFERENCES `Site` (site_id),
  ADD CONSTRAINT Provide_ibfk_2 FOREIGN KEY (food_bank_id) REFERENCES
`Food_Bank` (food_bank_id) ON DELETE SET NULL,
  ADD CONSTRAINT Provide_ibfk_3 FOREIGN KEY (food_pantry_id) REFERENCES
`Food_Pantry` (food_pantry_id) ON DELETE SET NULL,
  ADD CONSTRAINT Provide_ibfk_4 FOREIGN KEY (soup_kitchen_id) REFERENCES
`Soup_Kitchen` (soup_kitchen_id) ON DELETE SET NULL,
  ADD CONSTRAINT Provide_ibfk_5 FOREIGN KEY (shelter_id) REFERENCES `Shelter`
(shelter_id) ON DELETE SET NULL;

ALTER TABLE `Item`
  ADD CONSTRAINT Item_ibfk_1 FOREIGN KEY (food_bank_id) REFERENCES `Food_Bank`
(food_bank_id) ON DELETE SET NULL;

ALTER TABLE `Request`
  ADD CONSTRAINT Request_ibfk_1 FOREIGN KEY (user_email) REFERENCES `User`
(user_email),
  ADD CONSTRAINT Request_ibfk_2 FOREIGN KEY (item_name) REFERENCES `Item`
(item_name);
```

```
ALTER TABLE `Room`
  ADD CONSTRAINT Room_ibfk_1 FOREIGN KEY (shelter_id) REFERENCES `Shelter`
(shelter_id) ON DELETE CASCADE;

ALTER TABLE `Bunk`
  ADD CONSTRAINT Bunk_ibfk_1 FOREIGN KEY (shelter_id) REFERENCES `Shelter`
(shelter_id) ON DELETE CASCADE;

ALTER TABLE `Waitlist`
  ADD CONSTRAINT Waitlist_ibfk_1 FOREIGN KEY (room_number) REFERENCES `Room`
(room_number),
  ADD CONSTRAINT Waitlist_ibfk_2 FOREIGN KEY (client_id) REFERENCES `Client`
(client_id),
  ADD CONSTRAINT Waitlist_ibfk_3 FOREIGN KEY (shelter_id) REFERENCES `Shelter`
(shelter_id) ON DELETE CASCADE;

ALTER TABLE `Log_Entry`
  ADD CONSTRAINT Log_Entry_ibfk_1 FOREIGN KEY (client_id) REFERENCES `Client`
(client_id);
```

# Appendix 1: SQL test code

```
USE cs6400_sp17_team073;
/*INSERT INTO Site values(1,"s","s","c","tx","site",78759,"numbers");
INSERT INTO User values("Taylor", "TaylorPhebus@gmail.com","password","Taylor", 1);
INSERT INTO Request(username,item_name, request_status,units_requested,units_fulfilled)
VALUES ("Taylor","Peanut Butter","pending",10,0);
INSERT INTO Food_Bank(description_string) VALUES("Food Bank");*/

/*Log in
SELECT count(username) FROM User WHERE username=$username and
password=$password;
*/
SELECT count(username) FROM User WHERE username="taylor" and password="password";

/*View Site Data
SELECT Site.short_name, Site.street_address, Site.city, Site.state, Site.full_name, Site.zip,
Site.contact_number,Food_Pantry.description_string, Food_Pantry.hours,
Food_Pantry.conditions_for_use, Food_Bank.description_string,
Soup_Kitchen.description_string, Soup_Kitchen.hours, Soup_Kitchen.conditions_for_use,
Soup_Kitchen.available_seats, Shelter.description_string, Shelter.hours,
Shelter.conditions_for_use, Shelter.available_bunks, Shelter.available_rooms FROM Site LEFT
JOIN Provide on Provide.site_id=Site.site_id LEFT JOIN Food_Pantry on
Food_Pantry.food_pantry_id=Provide.food_pantry_id LEFT JOIN Food_Bank on
Food_Bank.food_bank_id=Provide.food_bank_id LEFT JOIN Soup_Kitchen on
Soup_Kitchen.soup_kitchen_id=Provide.soup_kitchen_id LEFT JOIN Shelter on
Shelter.shelter_id=Provide.shelter_id WHERE Site.site_id=$shelter_id;
*/
SELECT Site.short_name, Site.street_address, Site.city, Site.state, Site.full_name, Site.zip,
Site.contact_number,Food_Pantry.description_string, Food_Pantry.hours,
Food_Pantry.conditions_for_use, Food_Bank.description_string,
Soup_Kitchen.description_string, Soup_Kitchen.hours, Soup_Kitchen.conditions_for_use,
Soup_Kitchen.available_seats, Shelter.description_string, Shelter.hours,
Shelter.conditions_for_use, Shelter.available_bunks, Shelter.available_rooms FROM Site LEFT
JOIN Provide on Provide.site_id=Site.site_id LEFT JOIN Food_Pantry on
Food_Pantry.food_pantry_id=Provide.food_pantry_id LEFT JOIN Food_Bank on
Food_Bank.food_bank_id=Provide.food_bank_id LEFT JOIN Soup_Kitchen on
Soup_Kitchen.soup_kitchen_id=Provide.soup_kitchen_id LEFT JOIN Shelter on
Shelter.shelter_id=Provide.shelter_id WHERE Site.site_id=1;

/*Edit Site Data
UPDATE Site set
short_name=$short_name,street_address=$street_addr,city=$city,state=$state,full_name=$na
me,zip=$zip,contact_number=$contact WHERE site_id=$site_id;
*/
UPDATE Site set
short_name="tmp",street_address="addr",city="city",state="st",full_name="name",zip=12345,co
ntact_number="12345" WHERE site_id=1;
/*Make sure User table is populated, one time*/
/*Create Request
INSERT INTO Request(username,item_name, request_status,units_requested,units_fulfilled)
VALUES ($username,$requested_item,"pending",$requested_count,0);
*/

/*Edit Request
UPDATE Request SET Request.units_requested=5 WHERE
Request.username=$requestingUser AND Request.item_name=$item_name;
*/
UPDATE Request SET Request.units_requested=5 WHERE Request.username="Taylor" AND
Request.item_name="Peanut Butter";

/*Approve Request
```

```
UPDATE Request INNER JOIN Item on Request.item_name=Item.item_name INNER JOIN
User on Item.food_bank_id=User.site_id SET Request.units_fulfilled=3,
Request.request_status="approved" WHERE User.username=$username AND
Request.username=$requestingUser AND Request.item_name=$itemName;
*/
UPDATE Request INNER JOIN Item on Request.item_name=Item.item_name INNER JOIN
User on Item.food_bank_id=User.site_id SET Request.units_fulfilled=3,
Request.request_status="approved" WHERE User.username="Taylor" AND
Request.username="Taylor" AND Request.item_name="Peanut Butter";

/*View Requests Outstanding
SELECT Request.username, Request.item_name, Request.request_status,
Request.units_requested, Request.units_fulfilled FROM Request INNER JOIN Item on
Request.item_name=Item.item_name INNER JOIN User on Item.food_bank_id=User.site_id
WHERE User.username=$username;
*/
SELECT Request.username, Request.item_name, Request.request_status,
Request.units_requested, Request.units_fulfilled FROM Request INNER JOIN Item on
Request.item_name=Item.item_name INNER JOIN User on Item.food_bank_id=User.site_id
WHERE User.username="Taylor";

/*View Request Status
SELECT username, item_name, request_status,units_requested,units_fulfilled FROM Request
WHERE username= $username ORDER BY request_status;
*/
SELECT username, item_name, request_status,units_requested,units_fulfilled FROM Request
WHERE username= "Taylor" ORDER BY request_status;

/*Search Clients
SELECT full_name, client_id, description_string, head_of_household FROM Client WHERE
full_name like "%$search_field%";
*/
SELECT full_name, client_id, description_string, head_of_household FROM Client WHERE
full_name like "%Taylor%";

/*View client data
SELECT full_name, description_string, head_of_household FROM Client WHERE
client_id=$client_id;
*/
SELECT full_name, description_string, head_of_household FROM Client WHERE client_id=1;

/*View Waitlist
```

```
SELECT client_id, position from Waitlist WHERE shelter_id=$shelter_number AND
room_number=$room_number ORDER BY position ASC;
*/
SELECT client_id, position from Waitlist WHERE shelter_id=1 AND room_number=1 ORDER
BY position ASC;

/*Delete Waitlist Item
DELETE FROM Waitlist WHERE position=$position and shelter_id=$shelter_id AND
room_number=$room_number;
UPDATE Waitlist SET position=position-1 WHERE position>$position AND
shelter_id=$shelter_id AND room_number=$room_number;
*/
DELETE FROM Waitlist WHERE position=3 and shelter_id=1 AND room_number=1;
UPDATE Waitlist SET position=position-1 WHERE position>3 AND shelter_id=1 AND
room_number=1;

/*Move Waitlist Item
UPDATE Waitlist SET position=position-1 WHERE position>$old_position AND
shelter_id=$shelter_id AND room_number=$room_number ;
UPDATE Waitlist SET position=position+1 WHERE position>$new_position AND
shelter_id=$shelter_id AND room_number=$room_number;
*/
UPDATE Waitlist SET position=position-1 WHERE position>3 AND shelter_id=1 AND
room_number=1 ;
UPDATE Waitlist SET position=position+1 WHERE position>5 AND shelter_id=1 AND
room_number=1;

/*Search Inventory Items
SELECT item_name, number_of_units, storage_type, item_type, food_category,
supply_category,expiration_date, food_bank_id FROM Item where ($expiration_Date="*" or
expiration_date=$expiration_date) AND ($stoarge_type="*" OR storage_type=$storage_type)
AND ($food_type="*" OR item_type=$food_type) AND ($food_category="*" OR
food_category=$food_category") AND($supply_category="*" OR
supply_category=$supply_category) AND item_name LIKE "%$item_name%";
*/
SELECT item_name, number_of_units, storage_type, item_type, food_category,
supply_category,expiration_date, food_bank_id FROM Item where ("*"="*" or
expiration_date='20170618 10:11:12 AM') AND ("*"="*" OR storage_type="dry") AND ("*"="*"
OR item_type="food") AND ("*"="*" OR food_category="Pasta") AND("*"="*" OR
supply_category="N/A") AND item_name LIKE "%Peanut%";

/*Change Inventory Count
UPDATE Item SET number_of_units=$NUM_UNITS WHERE item_name=$ITEM_NAME;
```

```
*/
UPDATE Item SET number_of_units=4 WHERE item_name="Peanut Butter";

/*Add Inventory Item*/
/*Get item categories*/
/*Set up food bank to be sure there is one
SELECT distinct(item_type) FROM Item;
SELECT distinct(food_category) FROM Item;

INSERT INTO Item(item_name, number_of_units, storage_type, item_type,
food_category,supply_category, expiration_date, food_bank_id) VALUES ($ITEM_NAME,
$NUM_UNITS, $STORAGE_TYPE, $ITEM_TYPE, $FOOD_CATEGORY,
$SUPPLY_CATEGORY, $EXPIRATION_DATE, $FOOD_BANK_ID);
*/
SELECT distinct(item_type) FROM Item;
SELECT distinct(food_category) FROM Item;
/*INSERT INTO Item(item_name, number_of_units, storage_type, item_type,
food_category,supply_category, expiration_date, food_bank_id) VALUES ("Peanut
Butter",3,"cool","food","Tasty","N/A",'20170618 10:11:12 AM', 1);  Only works one time because
of duplicate key*/
/*View Meals*/
/*Get the total number of meals, and value of whatever we're lowest on
SELECT min(counts.count) as low, max(counts.count) AS total_meals FROM (
SELECT 'vegetable' AS type, count(item_name) AS count FROM Item WHERE food_category =
'vegetables'
UNION
SELECT 'mineral' AS type, count(item_name) AS count FROM Item WHERE food_category =
'beans' OR food_category = 'nuts' OR food_category = 'grains'
UNION
SELECT 'animal' AS type, count(item_name) AS count FROM Item WHERE food_category =
'meat' OR food_category = 'seafood' OR food_category = 'dairy') AS counts;
*/
SELECT min(counts.count) as low, max(counts.count) AS total_meals FROM (
SELECT 'vegetable' AS type, count(item_name) AS count FROM Item WHERE food_category =
'vegetables'
UNION
SELECT 'mineral' AS type, count(item_name) AS count FROM Item WHERE food_category =
'beans' OR food_category = 'nuts' OR food_category = 'grains'
UNION
SELECT 'animal' AS type, count(item_name) AS count FROM Item WHERE food_category =
'meat' OR food_category = 'seafood' OR food_category = 'dairy') AS counts;

/*Get the category we're lowest on to request more of
```

```
SELECT counts.type as send_more FROM (
SELECT 'vegetable' AS type, count(item_name) AS count FROM Item WHERE food_category =
'vegetables'
UNION
SELECT 'mineral' AS type, count(item_name) AS count FROM Item WHERE food_category =
'beans' OR food_category = 'nuts' OR food_category = 'grains'
UNION
SELECT 'animal' AS type, count(item_name) AS count FROM Item WHERE food_category =
'meat' OR food_category = 'seafood' OR food_category = 'dairy') AS counts WHERE
counts.count=low;
*/
SELECT counts.type as send_more FROM (
SELECT 'vegetable' AS type, count(item_name) AS count FROM Item WHERE food_category =
'vegetables'
UNION
SELECT 'mineral' AS type, count(item_name) AS count FROM Item WHERE food_category =
'beans' OR food_category = 'nuts' OR food_category = 'grains'
UNION
SELECT 'animal' AS type, count(item_name) AS count FROM Item WHERE food_category =
'meat' OR food_category = 'seafood' OR food_category = 'dairy') AS counts WHERE
counts.count=2;

/*View bunks/rooms*
SELECT count(Bunk.bunk_number), bunk_type, Shelter.description_string, Shelter.hours,
Shelter.conditions_for_use FROM Bunk INNER JOIN Shelter on
Shelter.shelter_id=Bunk.shelter_id GROUP BY Bunk.bunk_type, Shelter.shelter_id
*/
SELECT count(Bunk.bunk_number), bunk_type, Shelter.description_string, Shelter.hours,
Shelter.conditions_for_use FROM Bunk INNER JOIN Shelter on
Shelter.shelter_id=Bunk.shelter_id GROUP BY Bunk.bunk_type, Shelter.shelter_id;
/*Add Client
Insert into Client(full_name, description_string, head_of_household)
VALUES($FullName,$Description,$HeadOfHousehold);
*/
Insert into Client(full_name, description_string, head_of_household)
VALUES("Taylor","Smart",True);

/*Check in client to service
SELECT Shelter.shelter_id, Bunk.bunk_number from Shelter INNER JOIN User on
User.site_id=Shelter.shelter_id INNER JOIN Bunk on Bunk.shelter_id=Shelter.shelter_id
WHERE Shelter.available_bunks>0 AND User.username = $username AND
Bunk.bunk_type=$gender OR Bunk.bunk_type="mixed" ORDER BY Bunk.bunk_type LIMIT 1;
```

```
UPDATE Shelter INNER JOIN User on User.site_id=Shelter.shelter_id INNER JOIN Bunk on
Bunk.shelter_id=Shelter.shelter_id SET
Shelter.available_rooms=Shelter.available_rooms-1,Bunk.occupied=True WHERE
Shelter.shelter_id=0 AND Bunk.bunk_number=0;
*/
SELECT Shelter.shelter_id, Bunk.bunk_number from Shelter INNER JOIN User on
User.site_id=Shelter.shelter_id INNER JOIN Bunk on Bunk.shelter_id=Shelter.shelter_id
WHERE Shelter.available_bunks>0 AND User.username = "taylor" AND
Bunk.bunk_type="female" OR Bunk.bunk_type="mixed" ORDER BY Bunk.bunk_type LIMIT 1;

UPDATE Shelter INNER JOIN User on User.site_id=Shelter.shelter_id INNER JOIN Bunk on
Bunk.shelter_id=Shelter.shelter_id SET
Shelter.available_rooms=Shelter.available_rooms-1,Bunk.occupied=True WHERE
Shelter.shelter_id=0 AND Bunk.bunk_number=0;

/*Modify bunk count
Update Shelter INNER JOIN User ON User.site_id=Shelter.shelter_id SET
Shelter.available_bunks=$available_bunks WHERE User.username=$username
*/

Update Shelter INNER JOIN User ON User.site_id=Shelter.shelter_id SET
Shelter.available_bunks=3 WHERE User.username="Taylor"
```